Technical Report - Spam Detection

Lennart Kasserra

Link to Repository

Introduction

Spam emails are a very common and well-known nuisance: they often contain phishing attempts, malware, and scams that can compromise personal data and financial security. While traditional rule-based filters have long struggled to adapt to evolving spam tactics, machine learning models can dynamically learn patterns from data and are thus more robust. The goal of this project was to build a model that could reliably classify emails as either "spam" or "no spam" ("binary classification task"), and to evaluate whether a deep learning approach was preferrable to using "traditional" models. The data was provided via the UC Irvine machine learning repository.

Overall, my findings indicate that a deep learning approach is not neccessary: a properly tuned random forest classifier outperformed a deep neural network on all evaluated metrics.

Analysis

The data contained 57 features, and one column denoting the class label (either spam or no spam). Most features were related to the presence and frequency of certain words or special characters, or excessive capitalization. I suspected (1) class imbalance, and (2) high feature correlations, which is what I decided to investigate. I also ran a principal component analysis (PCA) to see if we could find some patterns in a transformed representation of the data.

After investigation, I decided on my preprocessing steps: despite class imbalance not being too bad, I decided to use synthetic minority class oversampling (SMOTE) to generate synthetic examples of spam emails and equalize the number of training examples per class. I further log-transformed and normalized all features, dropped highly correlated features ($r > 0.9$ or $r < -0.9$) and features with near-zero variance.

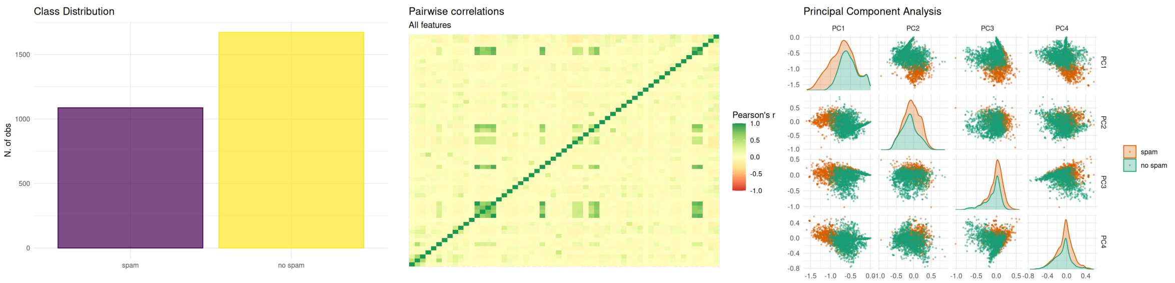


Figure 1: Exploratory data analysis: investigating class labels, feature correlations & a principal component analysis (training set only; training-validation-testing split was stratified on class label).

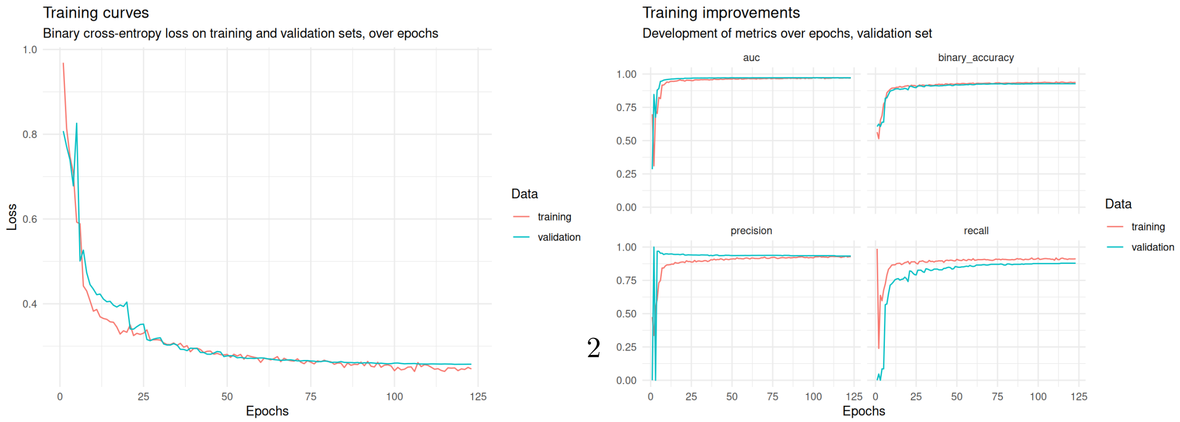
Methods

The main model is a deep neural network with four hidden layers. The hidden layers had 128, 64, 32 & 16 neurons respectively, all using the Rectified Linear Unit (ReLU) activation function and with dropout applied after every layer. The output layer was a one-neuron layer with sigmoid activation function, appropriate for this sort of binary classification task. Hyperparameters are shown in the table below.

Hyperparameter	Value
Learning rate	0.001 (scheduled)
Optimizer	Adam
Dropout Rate	0.25
Batch size	32
Epochs	max. 250 (early stopping)
L2 Regularization	0.001

Figure 2: Hyperparameters

Learning rate scheduling was set up to reduce the learning rate by a factor of 0.8 after three epochs of patience[1] (“Reduce on Pleateau”). Early stopping would kick in after validation loss has not decreased for five epochs in a row. Binary cross-entropy loss was used as the loss function for training.



(a) Loss curves

(b) Training curves for different metrics

Figure 3: Training process of the deep neural network.

[1] I.e. after validation performance did not increase for three successive epochs. This is meant to stabilize training & help the model settle into minima.

To select a proper competitor to compare the deep neural network against, I evaluated three "traditional" models: a penalized logistic regression model, a gaussian naive bayes classifier, and a random forest. For all models, I tuned their hyperparameters using regular grid search with ten-fold cross validation [2], and then compared their performance on the training set to that of the neural network. Results can be found in figure 4.

I decided that for the given application - detecting spam - false positives (i.e. falsely labeling genuine emails as spam, and thus potentially removing them from the inbox) were more costly or undesirable than the annoyance of having an occasional spam email slip through, and thus decided that precision ("what proportion of emails classified as spam were actually spam") was the most crucial metric.

The random forest classifier performed roughly on par with the deep neural network on the training set when considering both precision but also accuracy, so I decided to use this model as competitor to the neural network.

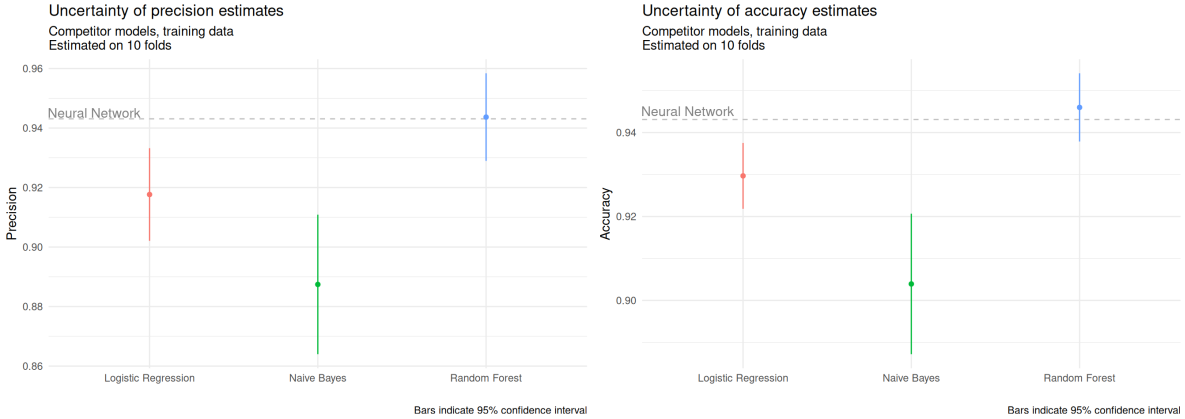


Figure 4: Potential competitors versus the deep neural network, evaluated on the training set. Confidence intervals were calculated across the ten folds used for tuning.

Results

The deep neural network was outperformed by a tuned Random Forest model on all metrics when evaluated on the test set. Crucially, it delivered lower precision than the Random Forest, but also fell behind on accuracy and f1-score. While the performance differences are at best marginal, when combined with the explainability that Random Forest offers over a deep neural network this is a solid case for using a "traditional" over a deep learning approach. Below are

[2] Except for the gaussian naive bayes, where I did not tune the smoothing parameter & just used raw probabilities instead (the smoothing should not have any effect in this setting).

the metrics computed on the test set, both as point estimates and with bootstrapped 95% confidence intervals.

Model	Precision	Accuracy	F1
Neural Network	0.932	0.939	0.922
Random Forest	**0.935**	**0.942**	**0.926**

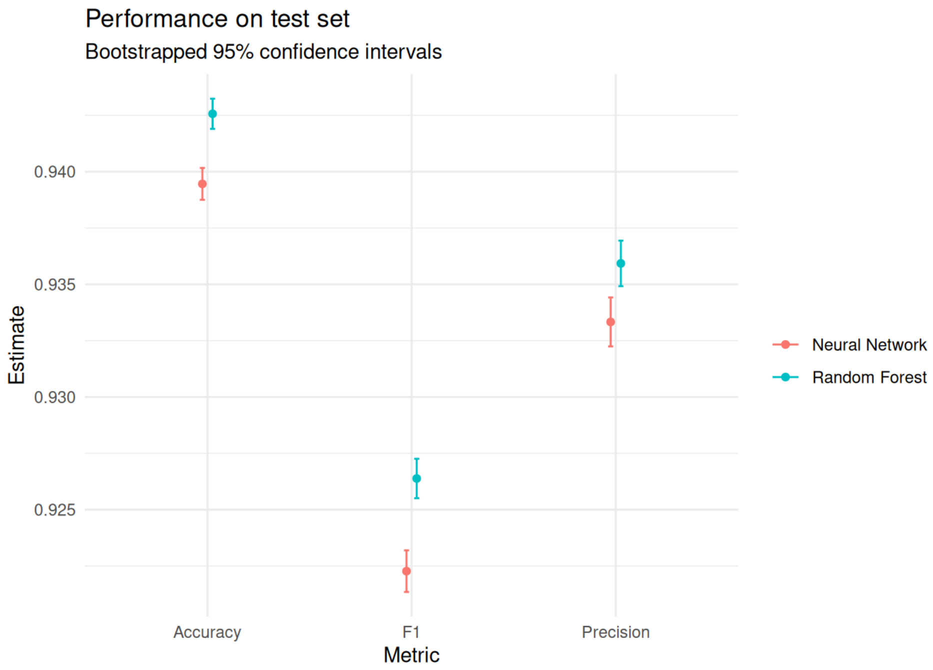
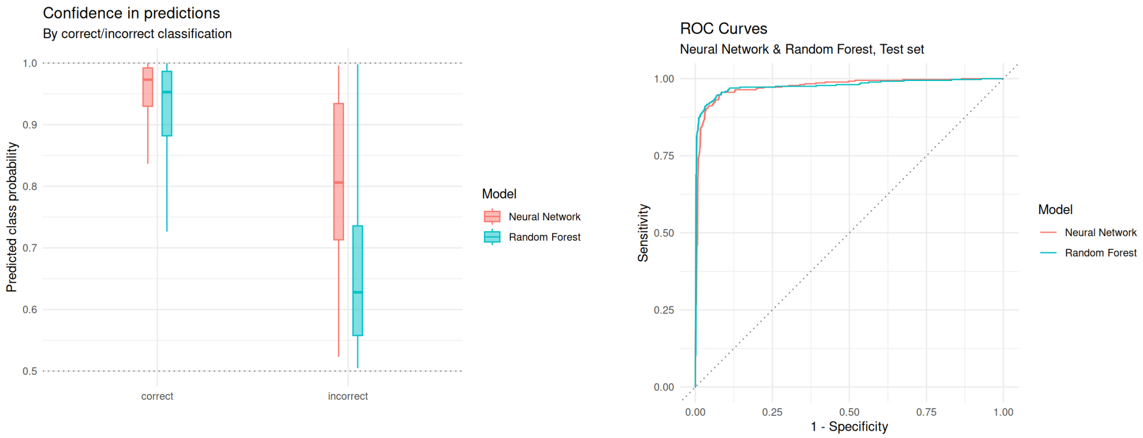


Figure 5: Test metrics with bootstrapped 95% confidence intervals, computed across 500 resamples.

Additionally, I examined the models' "confidence" in their classifications by comparing their outputted class probabilities when being correct versus when being wrong. On average, the neural network is also overconfident in misclassifications, which might hint at a deeper structural problem. However, examining the ROC curves shows just how close together the two models are in terms of performance.

Over all, a deep learning approach is not neccessary for this use case, and given this data.



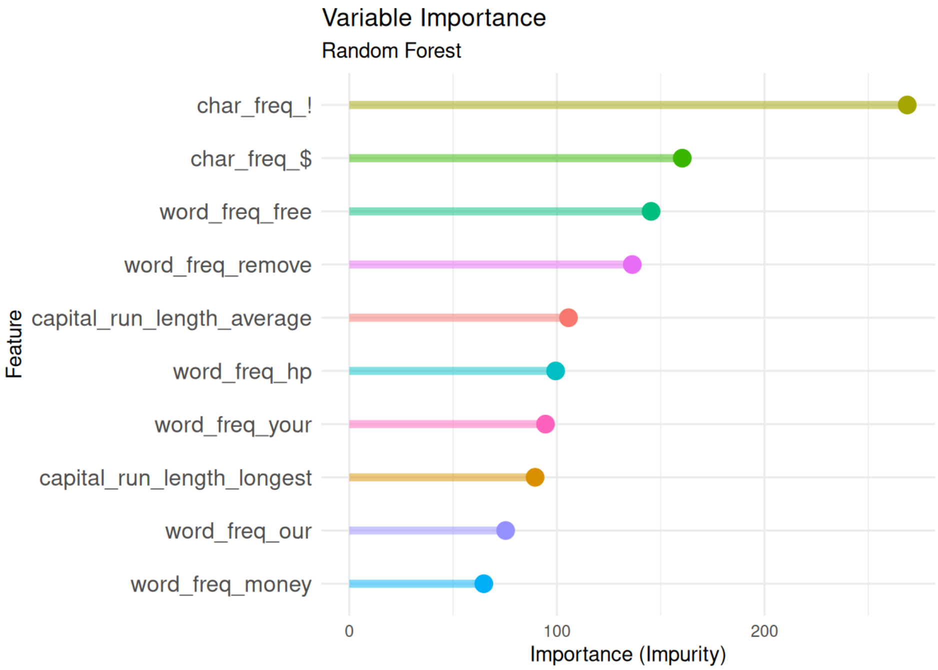
(a) Model confidence: predicted class probabilities for right and wrong classifications.

(b) Receiver-operating characteristic (ROC) curves.

Figure 6: Confidence & Test-set ROC curves.

Reflection

Given we decided on the Random Forest model, we can try to generate some explicit knowledge about our data. We can do this by examining variable importance, which I kept track of when building the model.



It seems that the number of exclamation marks and dollar signs, as well as promising free stuff are a giveaway about the true label of an observation. This is perhaps not a groundbreaking insight, but nevertheless illustrates once more the value of “traditional” models vis-a-vis deep learning approaches, where generating explicit knowledge about the data and classes is much less straight-forward, if at all possible.

Regarding possible improvements, it would have been tempting to try out more “traditional” models, and to explore a larger search space when tuning them, as it seems that when properly tuned they may outperform deep learning models on this task.

It might also have been fruitful to explore how smaller or more shallow neural networks compare to the deep model.